Chemometrics and Intelligent Laboratory Systems, 19 (1993) 337–341
Elsevier Science Publishers B.V., Amsterdam

■ Software Description

CFIT: a genetic algorithm for survival of the fitting

C.B. Lucasius, A.P. de Weijer, L.M.C. Buydens and G. Kateman

Laboratory for Analytical Chemistry, Faculty of Science, Katholieke Universiteit Nijmegen, Toernooiveld 1, 6525 ED Nijmegen (Netherlands)

(Received 12 November 1992; accepted 9 December 1992)

Abstract

Lucasius, C.B., De Weijer, A.P., Buydens, L.M.C. and Kateman, G., 1993. CFIT: a genetic algorithm for survival of the fitting. *Chemometrics and Intelligent Laboratory Systems*, 19: 337–341.

CFIT is discussed: an application program of a genetic algorithm dedicated to spectral curve fitting. CFIT's practical utility resides in several attractive properties of genetic algorithms that other techniques traditionally used for curve fitting apparently lack, inter alia robustness. Thus far, the results are promising. Since recently, CFIT is used at a chemical company.

INTRODUCTION

(The work presented here is an extension of [1].)

Peak deconvolution by curve fitting is an important technique among data reduction techniques used for the interpretation of experimental spectra. The procedure is based on a mathematical model assumed for the peaks in the spectra; we adopted a Pearson VII peak model, which encompasses, inter alia, pure Gauss forms, pure Lorentz forms and mixed forms thereof. Each peak is described by four model parameters — A_0 (height), X_0 (position), H (width at half height) and Z (tailing factor); the latter determines the Gauss/Lorentz proportion. The mathematical model is non-linear and since it cannot be solved analytically, a search procedure is required: iteratively, a set of values for the model parameters is proposed and in each step the model is used to calculate a spectrum that is compared with the known experimental spectrum; the comparison is done on the basis of some predefined dissimilarity criterion, or error function.

Before starting a curve fitting procedure, the user must specify the maximum number of peaks expected in the experimental spectrum. Evidently, this is a critical decision; in general, one stays at the safe side if the maximum is chosen too large (rather than too small), because this leaves open the possibility that zero heights are assigned to the excess peaks; on the other hand, the search is then complicated by ambiguities.

Correspondence to: Dr. L.M.C. Buydens, Laboratory for Analytical Chemistry, Faculty of Science, Katholieke Universiteit Nijmegen, Toernooiveld 1, 6525 ED Nijmegen, Netherlands.

SEARCH METHOD [2,3]

All search techniques have in common that they basically aim to improve an initial estimate of the true solution. Our program CFIT (Curve FITter) uses a genetic algorithm to search the global minimum on the error landscape in the space spanned by the model parameters; the global minimum represents the true solution. Best performance is obtained when a proposed set of values for the model parameters is encoded as a binary string, or a bitstring. A population of bitstrings is maintained and updated iteratively. (The initial population is chosen randomly.) In each step, all bitstrings are evaluated by the error function and are assigned a reverse error value as a measure of performance, or fitness (in genetic jargon).

The updating procedure is inspired by principles of natural evolution according to Darwin. First, the strings are reproduced at rates proportional to their fitness, until the new population thus formed reaches the size of the current population; the idea behind this is Darwin's 'survival of the fittest': the new population is, on average, expected to perform better due to the selection 'pressure' exerted; in CFIT this comes down to survival of the bitstrings that give rise to the best fitting spectra calculated. Next, in order to reach potentially better estimates, the bitstrings in the new population are modified to some controlled extent by operators reminiscent of crossover and mutation as applied to biological chromosomes; these and other genetic operators are based on random events. The evolution cycle is closed by replacing the current population with the new population, and the next cycle is started.

EVALUATION

Each bitstring evaluation proceeds in basically the following steps.

First, the bitstring under consideration is decoded into a string of real values for the unknown model parameters; this is a computationally cheap step. Then, applying the peak model to these values, datapoints are calculated that constitute a proposed spectrum; these data points are for the same wavelengths as the data points that constitute the experimental spectrum.

Spectrum calculations amount by far to the computationally most intensive part of CFIT; the burden (time), T_1, for one such calculation is roughly proportional to (1) N_{peak}, the number of peaks assumed (because peaks are calculated separately before they are summed into a spectrum), and (2) N_{data}, the number of data points that comprise the spectrum:

$$T_1 \propto N_{peak} N_{data}$$

Next, the error function compares the calculated spectrum with the experimental spectrum on the basis of a bi-criterion, comprising RMS (the root mean square — a measure of dissimilarity) and CORR (the correlation coefficient — a measure of similarity): simultaneously, RMS is minimized and CORR is maximized. Thereby, CORR is basically used to pronounce the RMS landscape; indeed, the bi-criterion usually leads to better search performance than when RMS alone is used.

PROPERTIES

The practical utility of CFIT resides in some generally attractive features of genetic algorithms: robustness (relative to local search) and efficiency (relative to global search).

Local search techniques traditionally applied to tackle curve fitting problems have a notorious record of unreliability, related to the following coincidence: (1) many local minima besides the global minimum may be present on the error landscape, especially when strong peak overlaps occur in the spectrum to be fit; (2) as a rule, no accurate initial estimate of the solution can be given. Consequently, the search ends up in the nearest optimum, which is frequently local; under these circumstances a short convergence time becomes worthless.

A more robust search strategy demands the use of search techniques that employ non-local search heuristics, e.g., statistical 'cooling' algorithms (simulated annealing) or genetic algorithms. In general, non-local search requires more

computation than local search in order to converge, but the result is more accurate and less sensitive to the choice of the initial estimate; besides, nowadays computation is fast and cheap.

Relative to global search — that is, systematic scan of the legally proposable solutions in the search space — genetic algorithms are extremely efficient. This is a consequence of exponential search behavior — a property that follows from the so-called schema theorem (the mathematical framework of genetic algorithms); the next section illustrates this point.

PERFORMANCE [3]

The efficiency of CFIT can be appreciated by the following comparison with global search.

Suppose the task is to fit five peaks ($N_{peak} = 5$); thus, the search space is spanned by twenty model parameters (four per peak). Next, the range of values for the model parameters is digitized into 512 levels; this reflects a reasonable desired precision. The levels define a search grid in the twenty-dimensional search space. The number of nodes on this grid is the size of the search space: $512^{20} \approx 10^{54}$. This size is prohibitively large for global search using today's or tomorrow's fastest computers. CFIT, on the other hand, requires only 10^4 evaluations to converge towards an acceptable solution; this corresponds to approximately 10 min real-time on an IBM PC 80486 platform (33 MHz) for a spectrum with $N_{data} = 200$.

CFIT's running time, T_{cfit}, equals $T_1 N_{popl} N_{iter}$, where N_{popl} is the population size (the number of bitstrings in the population) chosen and N_{iter} is the number of iterations (the number of population updates) required to attain convergence. Hence,

$$T_{cfit} \propto N_{peak} N_{data} N_{popl} N_{iter}$$

Empirically, we obtained best performance figures for $N_{popl} \approx 20 N_{peak}$; thereby we noted that, roughly, $N_{iter} \propto N_{peak}$. Thus, T_{cfit} increases roughly as the third power of N_{peak}. This compares very favorably with the exponential rate at which the size of the search space ($512^{4N_{peak}}$) grows with N_{peak}.

Nonetheless, when T_{cfit} would exceed practical time constraints, special provisions need to be made. Attractive strategies to that end are for example those wherein some method improves the initial estimate for CFIT first — thereby basically reducing N_{iter}, thus reducing T_{cfit}; a method used for such prior improvement is called a pre-hybridizer, and the overall sequential hybrid searching system is called a hyphenated search method. Its overall running time is $T_{hybr} = T_{pre} + T_{cfit}$, where T_{pre} is the time that the pre-hybridizer is run. The aim of pre-hybridization is, of course, to obtain a reduction in T_{cfit} that is larger than T_{pre}; the optimal T_{pre} must be determined empirically.

We have obtained good results with the following pre-hybridizers.

(1) An artificial neural network trained for finding peaks.

(2) CFIT itself, using a version of the experimental spectrum in which every second data point is weeded out; this halves the convergence time at the expense of some accuracy in the new estimate; a more severe weeding strategy may be considered in order to achieve further reduction at more expense of accuracy (incidentally, genetic algorithms are generally touted for their robustness against missing data and noise).

(3) CFIT itself, in one run applied to the left and another run applied to the right half of the spectrum, assuming half of the total amount of peaks in each; this reduces the total convergence time by roughly a factor $\frac{1}{2^3} + \frac{1}{2^3}$, at the expense of some accuracy in the new estimate; a higher degree of windowing (segmental partitioning) may be considered in order to achieve further reduction at more expense of accuracy.

(4) CFIT itself, halted before convergence is attained; the idea behind this is to use the estimate as the center of a pruned (smaller) search volume in a next run of CFIT (see example below).

CFIT comes with a post-hybridizer for further hyphenation (sequential hybridization): a steepest descent algorithm that serves to improve (refine) CFIT's ultimate estimate. Now, the overall running time of the hybrid searching system is $T_{hybr} = T_{pre} + T_{cfit} + T_{post}$, where T_{post} is the time that

the post-hybridizer is run. The post-hybridizer can play a similar role as the pre-hybridizer in minimizing T_{hybr}.

USAGE

CFIT is started with command-line arguments, e.g. (with explanation):

```
CFIT domain.1 domain.2  2 2 2 2 0
  1437 b
```

(see Table 1).

When the above session is configured for, say, 30 iterations, then a next session of CFIT might be specified as for example:

```
CFIT domain.2 domain.3  2 2 2 2 30
  9514 b
```

(Note that `domain.2` is now the first argument.) In this way, the estimate provided by the first CFIT run is further refined by CFIT itself (see the preceding section), now starting at logical iteration 30.

On start-up, CFIT implicitly assumes the existence of a file called `control.ga`, which specifies values for the genetic control parametes; e.g. the crossover rate, the mutation rate, the selection rate, etc.

SOFTWARE AND HARDWARE

The routines comprising CFIT are dichotomized into two parts: a domain-dependent part (concerning the problem representation and the error function) and a domain-independent part (concerning the genetic search heuristics). By 'domain-independent' routines are meant routines that can be used for other domains as well; for our application these were obtained from the software library GATES [4,5]. The integration of all routines resulted in CFIT.

CFIT is presently available for MS DOS on IBM PCs and for UNIX on SUN Sparc stations; a provisional manual is available. The source code of CFIT is written in C for efficiency and can be ported to any platform that supports an ANSI-standard C compiler.

Since recently, CFIT is used routinely at ARLA (Akzo Research Laboratories Arnhem, Netherlands). Depending on the response on the present paper, it is our intention to commercialize this software product; please contact the authors for more details.

REFERENCES

1 A.P. de Weijer, C.B. Lucasius, L. Buydens, G. Kateman, H.M. Heuvel and H. Mannee, A new approach to curve-fit-

TABLE 1

Example of CFIT command-line arguments and their explanation

`domain.1`	The name of an input file that contains information about the domain, inter alia: the name of the input file that contains the spectrum to be fit; an initial estimate of the solution; the dimensions of the search volume around this estimate; the name of an output file to which the fitted spectrum will be written
`domain.2`	The name of an output file with the same format as `domain.1` and that specifies, inter alia: the name of the input file that contains the spectrum that was fit; a new (hopefully better) estimate of the solution; the proposed pruned dimensions of the search volume around this estimate if `domain.2` would be used as the first argument in a next call of CFIT (see example)
`2 2 2 2`	Pruning factors for the search volume dimensions concerning the model parameter types A_0, X_0, H and Z, respectively
`0`	Logical iteration offset. The physical iteration offset is always 0, but the logical iteration offset should be set at the last iteration where a previous run ended, to ensure that entries written to output files that monitor performance are appended consecutively
`1437`	Seed for the random generator used
`b`	Batch mode (optional): interaction with the user is suppressed. This is especially useful when many sessions (runs) are planned in sequence, called from a batch file

ting using natural computation, *Analytical Chemistry*, submitted for publication.
2 C.B. Lucasius and G. Kateman, Understanding and using genetic algorithms. Part 1. Concepts, properties and context, *Chemometrics and Intelligent Laboratory Systems*, 19 (1993) 1–33.
3 C.B. Lucasius and G. Kateman, Understanding and using genetic algorithms. Part 2. Representation, configuration and hybridization, *Chemometrics and Intelligent Laboratory Systems*, submitted for publication.
4 C.B. Lucasius and G. Kateman, GATES towards evolutionary large-scale optimization: a software-oriented approach to genetic algorithms. Part 1. General perspective, *Computers & Chemistry*, submitted for publication.
5 C.B. Lucasius and G. Kateman, GATES towards evolutionary large-scale optimization: a software-oriented approach to genetic algorithms. Part 2. Toolbox description, *Computers & Chemistry*, submitted for publication.

This program has been implemented by us and performs as described. H.M. Heuvel, Akzo Research Laboratories Arnhem, P.O. Box 9300, 6800 SB Arnhem, Netherlands.